Uneven Seven Sided Dice Roller

Serkan Kıyaklı
Department of Electrical and
Electronics Engineering,
Abdullah Gül University
Kayseri, Türkiye
serkan.kiyakli@agu.edu.tr

Aliye Birgül
Department of Electrical and
Electronics Engineering,
Abdullah Gül University
Kayseri, Türkiye
aliye.birgul@agu.edu.tr

***Abstract*—This paper presents a model of a random generator circuit akin to an uneven seven-sided dice roller(Figure 1.1). The electronic circuit comprises a 1-bit-Data Generator(D_0, operating at 11.18khz), a clock source (400Hz, CLK1) sourced from a microcontroller(specifically, an Arduino), a 4-bit intermediate register(R_i) consisting of four D-flipflops, a 2-bit full adder composed of a transistor-level-1-bit adder with a C_o (carry in) of 1 and IC-level-1 bit adder, and a 3 bit-output register(R_o) comprised of D-flipflops with clock source(CLK4, 100Hz) that will be generated by processing CLK1. The Data generator utilizes a CMOS ring oscillator constructed with three cascaded inverters. To generate uneven outputs, as required for simulating the randomness of a seven-sided dice, the D_0 and clock source are intentionally operated asynchronously.**

Keywords—CMOS Ring oscillator, register, demultiplexer, full adder, Clock, Seven Sided Dice, uneven

I. Introduction

A random number generator is a creation to produce random data. It is commonly used in cryptography.[1] The text explains a design of uneven seven sided dice roller as an electrical circuit based model of random number generator. To produce information as logic values, the data, form D_0, is stored in R_i with CLK1. However the design of R_i is not a regular register design. D_0 must be distributed to and stored in R_i as shown in Figure 1.3.To ensure accuracy, it is acknowledged that some patterns may emerge due to the data not originating from purely random oscillations, and neither CLK1 nor D0 intersect with true randomness. After the data is stored in register, it is sent to a full adder with carry input is 1, so output of the adder will be in range of 1 to 7. Then the sum is sent to R_o synchronized with CLK4. After that last output is captured with a microcontroller, simply Arduino, then checked the randomness of the operation.

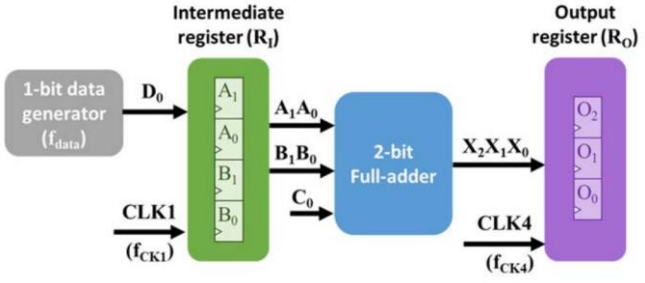

Fig. 1.1. Generator Representation

CLK1 is produced with Arduino with a simple code with tone function from its data pin 9. About CLK4, a specific design must be implemented to produce the signal by processing CLK1, and relation between CLK1 and CLK4 must be like in Figure1.2.

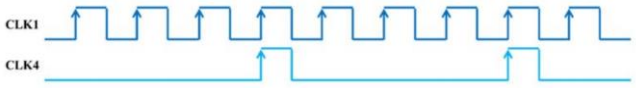

Fig. 1.2. CLK1 and CLK4

II. Design and Procedures

The scheme of the circuit implementation is designed first in LTSPICE. For the logic levels 5V is considered as logic 1 and 0V is considered as logic 0. The power is supplied from Arduino Uno 5V power pin. For the CMOS ring oscillator 2N7000 NMOS transistors and BS250 PMOS transistors are used. For the 1-bit-transistor level full adder BS170 NMOS transistors and ZVP0535A PMOS transistors are used.

CLK1	1st ↑	2nd ↑	3rd ↑	4th ↑	5th ↑	…
A_1	D_0	NC	NC	NC	D_0	…
A_0	NC	D_0	NC	NC	NC	…
B_1	NC	NC	D_0	NC	NC	…
B_0	NC	NC	NC	D_0	NC	…

Fig. 2.1. D_o distribution, NC means no change

A. CMOS Ring oscillator

A CMOS ring oscillator is a circuit that consists of cascaded odd numbers of CMOS inverter. Inverters changes the signals from low to high or high to low. The reason CMOS ring oscillator was chosen is simply because power consideration, due to low static current, and high noise margin. As operation works, between gates there is gate delay. The time it takes for the signal to travel through one

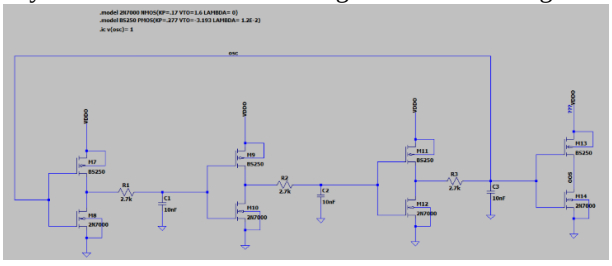

Fig. 2.2. CMOS Ring oscillator

inverter (from input to output) is known as the propagation delay of that inverter. When the power is turned on, the output of one inverter begins to propagate through the loop. Each inverter introduces a slight propagation delay due to the time that takes for the transistors to switch states. This delay is influenced by factors such as transistor characteristics, load capacitance, and RC part between inverters. The RC part creates a delay to reach the required amount of voltage to pass the threshold voltage of gates at certain time instances (Figure 1.4). The frequency calculation with the topology of circuit, it can be adjusted by changing capacitor and resistor values. The frequency formula [2],

$$f = \frac{1}{2nt} \quad (1)$$

n for the number of stages(inverters) and it is three in circuit Figure 1.4, the last inverter used after the ring oscillator to smooth output signal, and t is the propagation delay. t is equal to [3],

$$t = \frac{(t_{phl}+t_{plh})}{2} \quad (2)$$

t_{phl} and t_{plh} refer to the propagation delays associated with the transition of a signal from one logic state to another logic state in digital circuit. The frequency of D_0 is chosen 11kHz as theoretical value. Here is the propagation delay calculations as follows [4],

$$t_{phl} = \ln 2\ (R_n + R)(C_n + C) \quad (3)$$
$$t_{plh} = \ln 2\ (R_p + R)(C_p + C) \quad (4)$$

$R_n = 5\Omega$ and $R_p = 9\Omega$ are on resistance for NMOS and PMOS. $C_n = 20pF$ and $C_p = 20pF$ are input capacitances. Lastly C is the capacitance value RC part of the oscillator. As a result, the capacitor value for desired 11kHz frequency, is nearly 8nF, and for the purpose of simplicity in experimental part, 10nF capacitor is going to be used. In simulation, the frequency result is 10.89kHz and is remarkably close to calculated value, as can be seen in Figure 1.5.

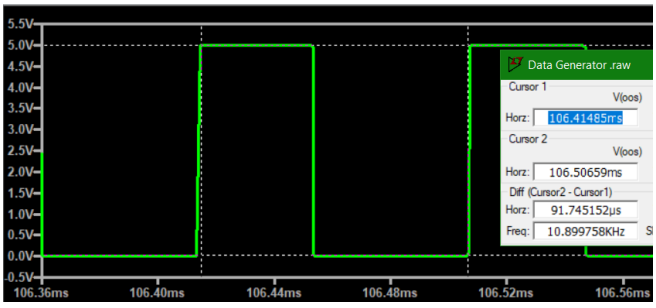

Fig. 2.3. Signals of Ring Oscillator

B. Intermediate Register

A register is a storage element to store number of data of bits, synchronized with a clock signal. For the seven sided dice roller circuit, there is a register, which is intermediate register, to store the data that comes from D_0. Intermediate register consists of four different D type positive-edge-triggered flipflops (A_1,A_0,B_1,B_0), and they use the CLK1 as clock source however, there is a specific way to store the data as can be seen in Figure 1.3. At the first-rising edge of CLK1 A_1 (first flipflop) is loaded, the other flipflops store their data. At the second-rising edge of CLK1 A_0 (second flipflop) is loaded and the other flipflops store their data. It is the same operation for B_1 at third-rising edge, for B_0 at fourth-rising edge then cycle repeats.

To do this, CLK1 is time-demultiplexed with a counter, which operates CLK1 as well, and with a 1x4 demux (Figure 1.6).

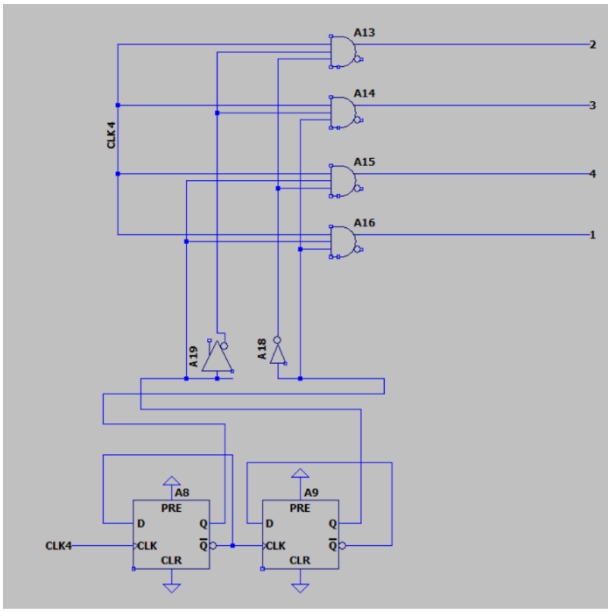

Fig. 2.4. Demultiplexer and Counter

The counter controls the control inputs of demux. CLK1 is the input of demux. So counter switching with time interval of 2.5ms and it is enough time for rising edges of CLK1 to pass through the demux. Here is the graph of output signals of demux,

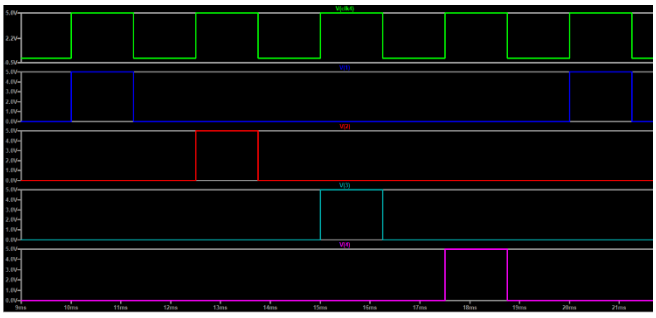
Fig. 2.5. Time-Demultiplexed clock signals

As can be understood now from Figure 1.7, the first signal, which is blue one, is sent to first flipflop(A_1), the rising edge will triggers A_1 and data is stored but the other's clock signals are 0, they store the data which is already in flipflops. Secondly, the second signal, which is red, is sent to A_0 and it is triggered and it gets a new data, while others stores their previous data because their clock source is 0. This operation continues as a cycle, and it can be seen after the purple signal blue signal comes and it will trigger again A_1.

For the register part, four different flipflops, which are positive edge triggered, is used. Here is schematic of the register,

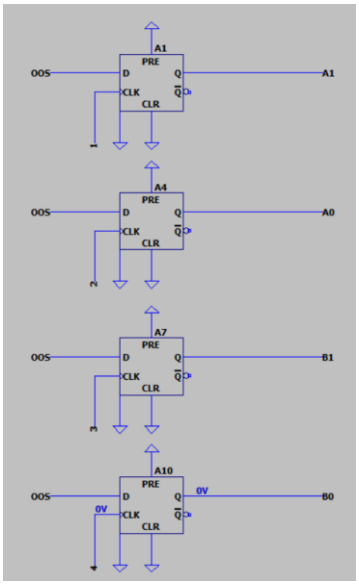

Fig. 2.6. Intermediate Register

When simulation is analyzed (Figure 1.9), it is easy to see that flipflops triggered asynchronously without affecting each other.

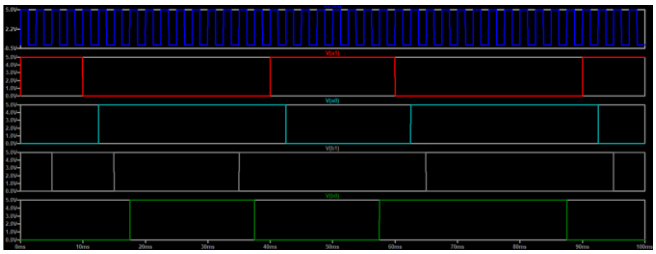

Fig. 2.7. Register outputs

From the result, the data for each flipflop must be stored 10ms because the cycle contains four clock period. Here is a magnified figure for outputs,

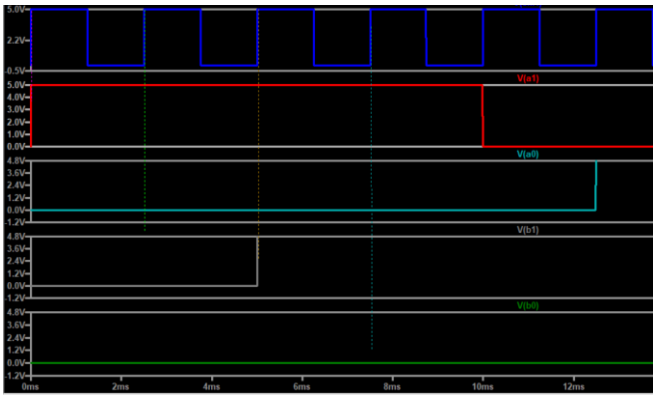

Fig. 2.8. Register outputs

C. *2-bit-Full Adder*

A full adder circuit is a crucial component which is used in computing to add three binary bits together, including two bits and a carry bit, C_0 from a previous addition. It produces sum bit and carry out bit as outputs. In Generator a 2-bit-full adder is used to add 4 bit number that all come from intermediate register output, C_o is 1 for this adder. It is considered that A_1A_0 and B_1B_0 are two different 2-bit numbers. One of these adders is implemented at transistor level with CMOS configuration. For designing circuitry, Boolean Algebra and truth table is used to simplify the full adder component.

Initially it is assumed that carry in C_o is 1 and within this consideration here is the truth table,

TABLE I. TRUTH TABLE OF FULL ADDER

IN_1	IN_2	SUM	C_OUT
0	0	1	0
0	1	0	1
1	0	0	1
1	1	1	1

Fig.2.9. Truth Table of Full adder When carry in is 1

It can be inferred from the table 1, SUM will become IN_1 XNOR IN_2, and C_OUT will become IN_1 OR IN_2. So the CMOS configuration of these operations here as follows,

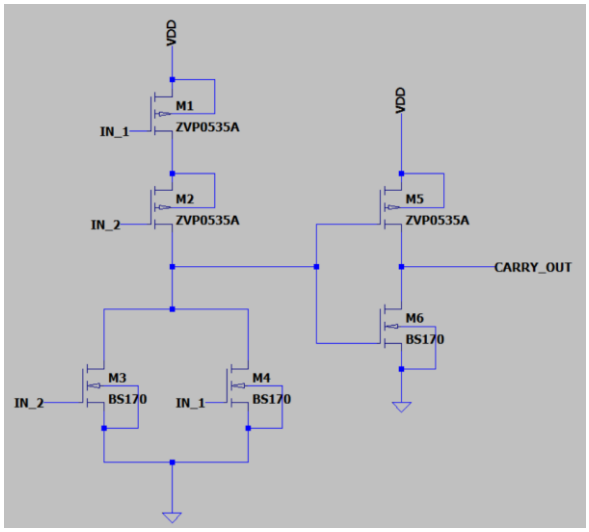

Fig. 2.10. CMOS Logic circuit of C_OUT

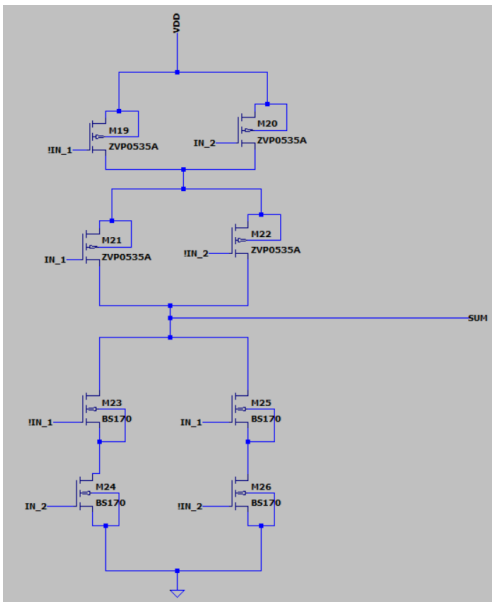

Fig. 2.11. CMOS Logic circuit of SUM

For the inverted inputs, 2 CMOS inverters was used (Figure 1.14.).

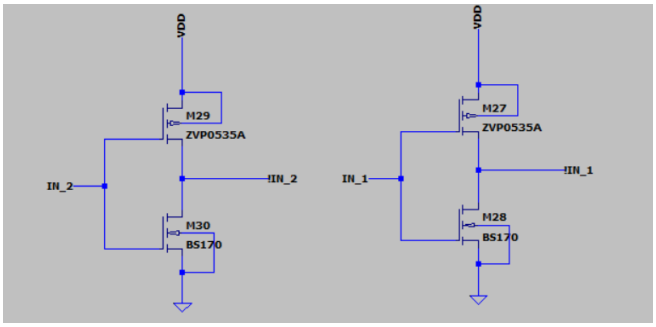

Fig. 2.12. Inverters of Input variables.

Second full adder is implemented at gate level with NAND configuration by using IC chips, additionally for this adder carry in is not settled as 1. Here is the configuration[5],

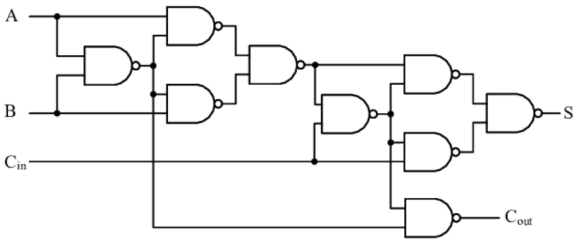

Fig. 2.13 NAND implementation of Second Full adder

After these implementations, register outputs A_1A_0 and B_1B_0 will be added. IN_1 become A_0 and IN_2 become B_0. In NAND implementation, A become A_1, B become B_1, and C_{in} become C_COUT. Here is the simulation result,

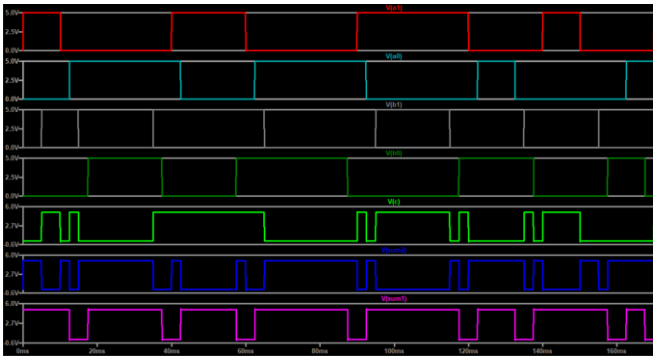

Fig. 2.14. Simulation result of 2-bit-full adder

D. Output Register

The 3 bit data, COUT SUM2 and SUM1, that comes from 2-bit-full adder is stored at the output register R_0 which is controlled by CLK4 in figure 1.2. To get CLK4, any output of demultiplexer circuit can be used. For the circuit part, 4. Pin is used to get CLK4(Figure 1.17.).

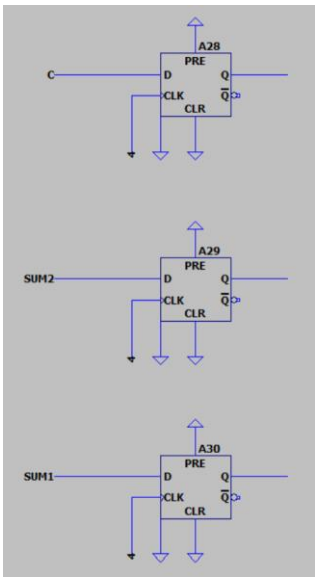

Fig 2.15. Output Register

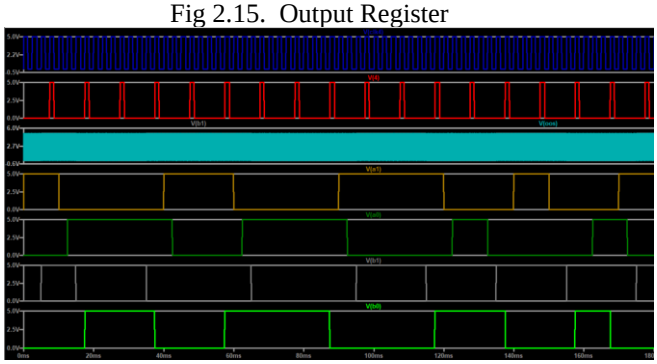

Fig 2.16 Full Timing Diagram of the System

III. Experiment And Result

The whole system was implemented on a several breadboards. (Figure 3.1)

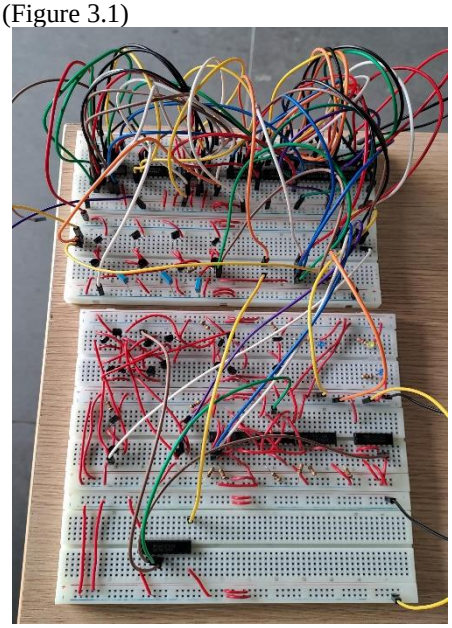

Fig. 3.1 Generator Implementation on breadboards

Respectively from top to bottom, first breadboard includes demux circuit and intermediate register, second includes CMOS ring oscillator, third and fourth is for 2-bit full adder,

The last one contains output register. Here is the output graph of intermediate register of circuit from a logic analyzer,

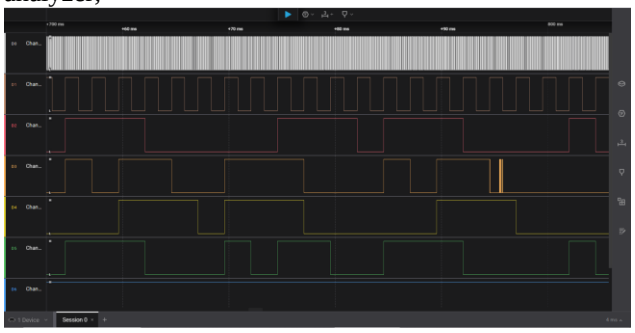

Fig. 3.2. Timing Diagram of Seven-sided Dice roller

For the adder implementation, here is the timing diagram of full adder,

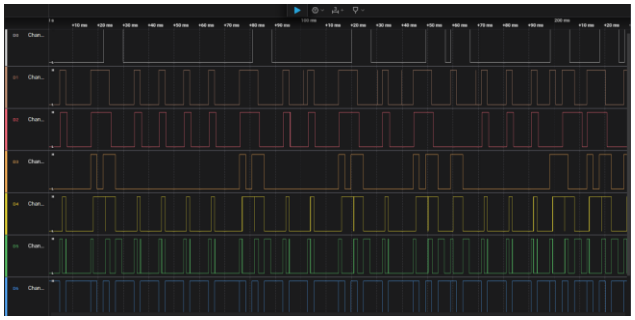

Fig. 3.3 Timing diagram of the output Full adder

In Figure 3.3 first four signal is intermediate register output (respectively A_1,A_0,B_1,B_0), other three signal is the output of full adder. (respectively COUT, SUM2, SUM1).Moreover, about output register, an Arduino Uno used to get bits occurrences and bits are recorded and analyzed. Here is the numbers in 10126 sample.

TABLE II. RESULT OF THE SYSTEM

Number	Occurrence	Probability
1	56	0.553
2	1172	11.574
3	1985	19.603
4	3680	36.3420
5	2800	27.65
6	355	3.505
7	78	0.77

IV. CONCLUSION

In conclusion, by utilizing logic circuits including 1-bit Data Generator (D0), a 4-bit intermediate register (Ri) , 2-bit full adder and a 3-bit-output register (Ro) parts, the target of the project was designing and implementing an Uneven Seven-Sided Dice Roller. The project commenced with the designing 1-bit Data Generator (D0) with implementing 3 stage inverters with CMOS configuration and the desired frequency range. This part served as the foundation for generating logic values 0 and 1, successfully. Produced logic values, were routed through a 4-bit intermediate register (Ri), including, counter for the demux inputs, and D-flip-flops. It served as storage component and before further processing, it ensured distribution of Do verify by the simulation results. For the full adder part, 1-bit full adder with transistor level and 1-bit full adder at gate level with IC chips were designed and simulated. Experimental and simulation outcomes were obtained effectively and correctly. The last step was passing the sums of the binary through 3-bit-output register (Ro) synchronized and controlled with CLK4 signal, facilitating that obtaining random numbers is easier. All in all, all steps were combined to the show function of the all system by using microcontroller Arduino. The distribution of the outcomes was examined with logic analyzer, and it was made a comparison, theoretically. By using a methodical approach, functional system of generating random numbers with the range of 0 to 7 is developed. Thanks to the project, various insights and information of logic system design is gained.

REFERENCES

[1] Jun & Kocher , THE INTEL ® RANDOM NUMBER GENERATOR https://www.rambus.com/wpcontent/uploads/2015/08/IntelRNG.pdf

[2] Pan, Ring Oscillator Frequency Measurements Using an Automated Parametric Test System https://download.tek.com/document/RingOscillatorWP.pdf

[3][4] Jaeger & Blalock, Microelectronic Circuit Design

[5] Abrishami, Mohammad Saeed & Pedram, Massoud & Nazarian, Shahin. (2019). CSM-NN: Current Source Model Based Logic Circuit Simulation - A Neural Network Approach. 393-400. 10.1109/ICCD46524.2019.00061.